ЛЕКЦИЯ №8. ЯЗЫК SQL. ОПРЕДЕЛЕНИЕ ДАННЫХ. СОЗДАНИЕ БАЗ ДАННЫХ. СОЗДАНИЕ ТАБЛИЦ. СВЕДЕНИЯ О ПРОМЫШЛЕННЫХ РЕЛЯЦИОННЫХ СУБД, ORACLE, ACCESS.

8.1 Объединения и стандарт SQL2

В стандарте SQL2 был определен совершенно новый метод поддержки внешних объединений, который не опирался ни на одну популярную СУБД. В спецификации стандарта SQL2 поддержка внешних объединений осуществлялась в предложении FROM с тщательно разработанным синтаксисом, позволявшим пользователю точно определить, как исходные таблицы должны быть объединены в запросе. Расширенное предложение FROM поддерживает также операцию UNION над таблицами и допускает сложные комбинации запросов на объединение операторов SELECT и объединений таблиц.

8.1.1 Внутренние объединения в стандарте SQL2

Две объединяемые таблицы соединяются явно посредством операции JOIN, а условие поиска, описывающее объединение, находится теперь в предложении ON внутри предложения FROM В условии поиска, следующем за ключевым словом ON, могут быть заданы любые критерии сравнения строк двух объединяемых таблиц.

Например: вывести список фамилий студентов, и названия групп, к которых они учаться.

```
SELECT StName, GrName
FROM Students INNER JOIN Groups
ON Students.GrNo = Groups.GrNo
```

(В этих простых двухтабличных объединениях все содержимое предложения WHERE просто перешло в предложение ON, и предложение ON не добавляет ничего нового в язык SQL.

Стандарт SQL2 допускает еще один вариант запроса на простое внутреннее объединение таблиц Students и Groups. Так как связанные столбцы этих таблиц имеют одинаковые имена и сравниваются на предмет равенства (что делается довольно часто), то можно использовать альтернативную форму предложения ON, в которой задается список имен связанных столбцов:

```
SELECT StName, GrName
FROM Students INNER JOIN Groups USING (GrNo)
```

Ниже приведен синтаксис оператора JOIN:

1. естественное соединение.

   FROM спецификация_таблиц,…,
   таблица1
   NATURAL {INNER|FULL[OUTER]|LEFT[OUTER]|RIGHT[OUTER]}JOIN
   таблица2 …

2. соединение с использованием выражения.

   FROM спецификация_таблицы,…,
   таблица1
   {INNER|[OUTER] FULL|[OUTER]LEFT|[OUTER]RIGHT} JOIN
   таблица2
   ON условие | USING(список_столбцов),…

3. объединение или декартово произведение.

FROM спецификация_таблицы,…,

таблица1 {UNION | CROSS JOIN} таблица2 ,…

Объединение двух таблиц, в котором связанные столбцы имеют идентичные имена, называется естественным объединением, так как обычно это действительно самый "естественный" способ объединения двух таблиц. Запрос на выборку пар фамилия студента/название группы, в которой он учиться, можно выразить как естественное объединение следующим образом:

SELECT StName, GrName

FROM Students NATURAL INNER JOIN Groups

8.1.2 Внешние объединения в стандарте SQL2

Стандарт SQL2 обеспечивает полную поддержку внешних объединений, расширяя языковые конструкции, используемые для внутренних объединений. Например, для построения таблицы подчиненности преподавателей можно применить следующий запрос:

SELECT Chief.TName, SubOrdinate.TName

FROM Teachers AS Chief FULL OUTER JOIN Teachers AS SubOrdinate

ON Chief.TNo = SubOrdinate.TChiefNo

Результат такого запроса (данные из Приложения А) приведен на рис. 0.1.

Chief.TName	SubOrdinate.TName
NULL	Иванов
Иванов	Петров
Петров	Стрельцов
Петров	Сидоров
Сидоров	NULL
Стрельцов	NULL

рис. 0.1 Результатом такого запроса на внешнее объединение.

Таблица результатов запроса будет содержать по одной строке для каждой связанной пары начальник/подчиненный, а также по одной строке для каждой несвязанной записи для начальника или подчиненного, расширенной значениями NULL в столбцах другой таблицы.

Ключевое слово OUTER, так же как и ключевое слово INNER, в стандарте SQL2 не является обязательным. Поэтому предыдущий запрос можно, было бы переписать следующим образом:

SELECT Chief.TName, SubOrdinate.TName

FROM Teachers AS Chief FULL JOIN Teachers AS SubOrdinate

ON Chief.TNo = SubOrdinate.TChiefNo

По слову FULL СУБД сама определяет, что запрашивается внешнее объединение.

Вполне естественно, что в стандарте SQL2 левое и правое внешние объединения обозначаются словами LEFT и RIGHT вместо слова FULL. Вот вариант того же запроса, определяющий левое внешнее объединение:

SELECT Chief.TName, SubOrdinate.TName

FROM Teachers AS Chief LEFT OUTER JOIN Teachers AS SubOrdinate

ON Chief.TNo = SubOrdinate.TChiefNo

В результате такого запроса (данные из Приложения А.) будет получено следующее отношение (рис. 0.2).

Chief.TName	SubOrdinate.TName
Иванов	Петров
Петров	Стрельцов
Петров	Сидоров
Сидоров	NULL
Стрельцов	NULL

рис. 0.2 Результатом такого запроса на внешнее объединение

8.1.3 Перекрестные объединения и запросы на объединение в SQL2

Расширенное предложение FROM в стандарте SQL2 поддерживает также два других способа соединения данных из двух таблиц – декартово произведение и запросы на объединение. Строго говоря, ни один из них не является операцией "объединения", но они поддерживаются в стандарте SQL2 с помощью тех же самых предложений, что и внутренние и внешние объединения. Вот запрос, создающий декартово произведение таблиц Students и Groups:

```
SELECT *
FROM Students CROSS JOIN Groups
```

8.1.4 Многотабличные объединения в стандарте SQL2

Одно из крупных преимуществ расширенного предложения FROM заключается в том, что оно дает единый стандарт для определения как внутренних и внешних объединений, так и произведений и запросов на объединение. Другим, даже еще более важным преимуществом этого предложения является то, что оно обеспечивает очень ясную и четкую спецификацию объединений трех и четырех таблиц, а также произведений и запросов на объединение. Для построения этих сложных объединений любые выражения описанные ранее, могут быть заключены в круглые скобки. Результирующее выражение, в свою очередь, можно использовать для создания других выражений объединения, как если бы оно было простой таблицей. Точно так же, как SQL позволяет с помощью круглых скобок комбинировать различные арифметические операции (+, –, * и /) и строить сложные выражения, стандарт SQL2 дает возможность создавать сложные выражения для объединений.

8.2 Итоговые запросы на чтение. Агрегатные функции

Для подведения итогов по информации, содержащейся в базе данных, в SQL предусмотрены агрегатные (статистические) функции. Агрегатная функция принимает в качестве аргумента какой-либо столбец данных целиком, а возвращает одно значение, которое определенным образом подытоживает этот столбец. Например, агрегатная функция AVG() принимает в качестве аргумента столбец чисел и вычисляет их среднее значение.

В SQL имеется шесть агрегатных функций, которые позволяют получать различные виды итоговой информации. Ниже описан синтаксис этих функций:

1. функция SUM() вычисляет сумму всех значений, содержащихся в столбце:

```
SUM(выражение | [DISTINCT] имя_столбца)
```

2. функция AVG() вычисляет среднее всех значений, содержащихся в столбце:

```
AVG(выражение | [DISTINCT] имя_столбца)
```

3. функция MIN() находит наименьшее среди всех значений, содержащихся в столбце:

```
MIN(выражение | имя_столбца)
```

4. функция MAX() находит наибольшее среди всех значений, содержащихся в столбце:

```
MAX(выражение | имя_столбца)
```

5. функция COUNT() подсчитывает количество значений, содержащихся в столбце:

```
COUNT([DISTINCT] имя_столбца)
```

6. функция COUNT(*) подсчитывает количество строк в таблице результатов запроса:

```
COUNT(*)
```

8.2.1 Агрегатные функции и значения NULL

В стандарте ANSI/ISO также определены следующие точные правила обработки значений NULL в агрегатных функциях:

1. если какие-либо из значений, содержащихся в столбце, равны NULL, при вычислении результата функции они исключаются;
2. если все значения в столбце равны NULL, то функции SUM(), AVG(), MIN() и MAX () возвращают значение NULL; функция COUNT () возвращает ноль;
3. если в столбце нет значений (т.е. столбец пустой), то функции SUM() , AVG(), MIN() и MAX() возвращают значение NULL; функция COUNT() возвращает ноль;
4. функция COUNT(*) подсчитывает количество строк и не зависит от наличия или отсутствия в столбце значений NULL; если строк в таблице нет, эта функция возвращает ноль.

8.2.2 Запросы с группировкой (предложение GROUP BY)

Итоговые запросы, о которых до сих пор шла речь в настоящей главе рассчитывают итоговые результаты на основании всех записей в таблице. Однако необходимость в таких вычислениях возникает достаточно редко, чаще бывает необходимо получать промежуточные итоги на основании групп записей в таблице. Эту возможность предоставляет предложение GROUP BY оператора SELECT.

Запрос, включающий в себя предложение GROUP BY, называется запросом с группировкой, поскольку он объединяет строки исходных таблиц в группы и для каждой группы строк генерирует одну строку таблицы результатов запроса. Столбцы, указанные в предложении GROUP BY, называются столбцами группировки, поскольку именно они определяют, по какому признаку строки делятся на группы. Например, получить список фамилий студентов и их средних оценок.

```
SELECT StName, AVG(Mark)
FROM Marks INNER JOIN Students USING(StNo)
GROUP BY StName
```

8.2.3 Несколько столбцов группировки

SQL позволяет группировать результаты запроса на основании двух или более столбцов. Например, получить список фамилий студентов и их средних оценок за каждый семестр.

```
SELECT StName, Semester, AVG(Mark)
FROM Marks INNER JOIN Students USING(StNo)
GROUP BY StName, Semester
```

8.2.4 Ограничения на запросы с группировкой

На запросы, в которых используется группировка, накладываются дополнительные ограничения. Столбцы с группировкой должны представлять собой реальные столбцы таблиц, перечисленных в предложении FROM. Нельзя группировать строки на основании значения вычисляемого выражения.

Кроме того, существуют ограничения на элементы списка возвращаемых столбцов. Все элементы этого списка должны иметь одно значение для каждой группы строк. Это означает, что возвращаемым столбцом может быть:

1. константа;
2. агрегатная функция, возвращающая одно значение для всех строк, входящих в группу;
3. столбец группировки, который, по определению, имеет одно и то же значение во всех строках группы;
4. выражение, включающее в себя перечисленные выше элементы.

На практике в список возвращаемых столбцов запроса с группировкой всегда входят столбец группировки и агрегатная функция. Если последняя не указана, значит, запрос можно более просто выразить с помощью ключевого слова DISTINCT без использования предложения GROUP BY. И наоборот, если не включить в результаты

запроса столбец группировки, вы не сможете определить, к какой группе относится каждая строка результатов.

8.2.5 Значения NULL в столбцах группировки

В стандарте ANSI определено, что два значения NULL в предложении GROUP BY равны.

Строки, имеющие значение NULL в одинаковых столбцах группировки и идентичные значения во всех остальных столбцах группировки, помещаются в одну группу.

8.2.6 Условия поиска групп (предложение HAVING)

Точно так же, как предложение WHERE используется для отбора отдельных строк, участвующих в запросе, предложение HAVING можно применить для отбора групп строк. Его формат соответствует формату предложения WHERE. Предложение HAVING состоит из ключевого слова HAVING, за которым следует условие поиска. Таким образом, данное предложение определяет условие поиска для групп.

8.2.7 Ограничения на условия поиска групп

Предложение HAVING используется для того, чтобы включать и исключать группы строк из результатов запроса, поэтому на используемое в нем условие поиска наложены такие же ограничения, как и на элементы списка возвращаемых столбцов.

8.2.8 Предложение HAVING без GROUP BY

Предложение HAVING почти всегда используется в сочетании с предложением GROUP BY, однако синтаксис оператора SELECT не требует этого. Если предложение HAVING используется без предложения GROUP BY, SQL рассматривает полные результаты запроса как одну группу. Другими словами, агрегатные функции, содержащиеся в предложении HAVING, применяются к одной и только одной группе, и эта группа состоит из всех строк. На практике предложение HAVING очень редко используется без соответствующего предложения GROUP BY.

8.3 Вложенные запросы

Вложенным (или подчиненным) запросом называется запрос, содержащийся в предложении WHERE или HAVING другого оператора SQL. Вложенные запросы позволяют естественным образом обрабатывать запросы, выраженные через результаты других запросов.

Чаще всего вложенные запросы указываются в предложении WHERE оператора SQL. Когда вложенный запрос содержится в данном предложении, он участвует в процессе отбора строк.

Например, вывести список фамилий студентов, средний балл которых выше 4,5:

```
SELECT StName
FROM Students
WHERE (SELECT AVG(Mark)
        FROM Marks
        WHERE Marks.StNo = Students.StNo) > 4.5
```

Вложенный запрос всегда заключается в круглые скобки, но по-прежнему сохраняет знакомую структуру оператора SELECT, содержащего предложение FROM и необязательные предложения WHERE, GROUP BY и HAVING. Структура этих предложений во вложенном запросе идентична их структуре в оператора SELECT; во вложенном запросе эти предложения выполняют свои обычные функции. Однако между вложенным запросом и оператором SELECT имеется ряд отличий:

1. Таблица результатов вложенного запроса всегда состоит из одного столбца. Это означает, что в предложении SELECT вложенного запроса всегда указывается один возвращаемый столбец.

2. Во вложенный запрос не может входить предложение ORDER BY. Результаты вложенного запроса используются только внутри главного запроса и для пользователя остаются невидимыми, поэтому нет смысла их сортировать.

3. Вложенный запрос не может быть запросом на объединение нескольких различных операторов SELECT; допускается использование только одного оператора SELECT.

4. Имена столбцов во вложенном запросе могут являться ссылками на столбцы таблиц главного запроса. Это так называемые внешние ссылки (ссылка на Students.StNo в предыдущем примере). Внешние ссылки необязательно должны присутствовать во вложенном запросе.

8.3.1 Условия поиска во вложенном запросе

Вложенный запрос всегда является частью условия поиска в предложении WHERE или HAVING. Ранее были рассмотрены простые условия поиска, которые могут использоваться в этих предложениях. Кроме того, в SQL имеются следующие условия поиска во вложенном запросе.

Сравнение с результатом вложенного запроса (=, <>, <, <=, >, >=). Сравнивает значение выражения с одним значением, возвращенным вложенным запросом. Эта проверка напоминает простое сравнение.

проверяемое_выражение | = | <> | < | <= | > | >= | вложенный_запрос

Сравнение с результатом вложенного запроса является модифицированной формой простого сравнения. Значение выражения сравнивается со значением, возвращенным вложенным запросом, и если условие сравнения выполняется, то проверка возвращает значение TRUE. Эта проверка используется для сравнения значения из проверяемой строки с одним значением, возвращенным вложенным запросом, как показано первом примере.

Проверка на принадлежность результатам вложенного запроса (IN). Проверка на принадлежность результатам вложенного запроса (ключевое слово IN) является видоизмененной формой простой проверки на членство в множестве.

проверяемое_выражение [NOT] IN вложенный_запрос

Одно значение сравнивается со столбцом данных, возвращенных вложенным запросом, и если это значение равно одному из значений в столбце, проверка возвращает TRUE. Данная проверка используется, когда необходимо сравнить значение из проверяемой строки с множеством значений, возвращенных вложенным запросом. Например, вывести всю информацию о студентах, учащихся в группах сназваниями, начинающимися на букву ″А″:

```
SELECT *
FROM Students
WHERE GrNo IN (SELECT GrNo FROM Groups WHERE GrName LIKE ‘A%’)
```

Проверка на существование (EXISTS). В результате проверки на существование (ключевое слово EXISTS) можно выяснить, содержится ли в таблице результатов вложенного запроса хотя бы одна строка. Аналогичной простой проверки не существует. Проверка на существование используется только с вложенными запросами.

[NOT] EXISTS вложенный_запрос

Например, вывести фамилии студентов, которые в 1-м семестре сдали хотябы одну дисциплину:

```
SELECT StName
FROM Students
WHERE EXISTS (SELECT *
        FROM Marks
        WHERE Marks.StNo = Students.StNo AND Semester = 1)
```

Многократное сравнение (ANY и ALL). В проверке IN выясняется, не равно ли некоторое значение одному из значений, содержащихся в столбце результатов вложенного запроса. В SQL имеется две разновидности многократного сравнения – ANY и ALL, расширяющие предыдущую проверку до уровня других операторов сравнения. В обеих проверках некоторое значение сравнивается со столбцом данных, возвращенным вложенным запросом.

проверяемое_выражение | = | <> | < | <= | > | >= | ANY | ALL вложенный_запрос

Проверка ANY. В проверке ANY используется один из шести операторов сравнения (=, <>, <, <=, >, >=) для того, чтобы сравнить одно проверяемое значение со столбцом данных, возвращенным вложенным запросом. Проверяемое значение поочередно сравнивается с каждым значением, содержащимся в столбце. Если любое из этих сравнений дает результат TRUE, то проверка ANY возвращает значение TRUE.

Например, вывести список фамилий студентов, получивших в первом семестре хотя бы одну отличную оценку.

```
SELECT StName
FROM Students
WHERE 5 = ANY (SELECT Mark
        FROM Marks
        WHERE Marks.StNo = Students.StNo AND Semester = 1)
```

Если вложенный запрос в проверке ANY не создает ни одной строки результата или если результаты содержат значения NULL, то в различных СУБД проверка ANY может выполняться по-разному. В стандарте ANSI/ISO для языка SQL содержатся подробные правила, определяющие результаты проверки ANY, когда проверяемое значение сравнивается со столбцом результатов вложенного запроса:

1. если вложенный запрос возвращает пустой столбец результатов, то проверка ANY имеет значение FALSE (в результате выполнения вложенного запроса не получено ни одного значения, для которого выполнялось бы условие сравнения).
2. если операция сравнения имеет значение TRUE хотя бы для одного значения в столбце, то проверка ANY возвращает значение TRUE (имеется некоторое значение, полученное вложенным запросом, для которого условие сравнения выполняется).
3. если операция сравнения имеет значение FALSE для всех значений в столбце, то проверка ANY возвращает значение FALSE (можно утверждать, что ни для одного значения, возвращенного вложенным запросом, условие сравнения не выполняется);
4. если операция сравнения не имеет значение TRUE ни для одного значения в столбце, но в нем имеется одно или несколько значений NULL то проверка ANY возвращает результат NULL. (В этой ситуации невозможно но с определенностью утверждать, существует ли полученное вложенным запросом значение, для которого выполняется условие сравнения; может быть, существует, а может и нет – все зависит от "настоящих" значений неизвестных данных.)

Проверка ALL. В проверке ALL, как и в проверке ANY, используется один из шести операторов (=, <>, <, <=, >, >=) для сравнения одного проверяемого значения со столбцом данных, возвращенным вложенным запросом. Проверяемое значение

поочередно сравнивается с каждым значением, содержащимся в столбце. Если все сравнения дают результат TRUE, то проверка ALL возвращает значение TRUE.

Например, вывести список фамилий студентов, получивших в первом семестре только удовлетворительные оценки.

```
SELECT StName
FROM Students
WHERE 3 = ALL (SELECT Mark
    FROM Marks
    WHERE Marks.StNo = Students.StNo AND Semester = 1)
```

Если вложенный запрос в проверке ALL не возвращает ни одной строки или если результаты запроса содержат значения NULL, то в различных СУБД проверка ALL может выполняться по-разному. В стандарте ANSI/ISO для языка SQL содержатся подробные правила, определяющие результаты проверки ALL, когда проверяемое значение сравнивается со столбцом результатов вложенного запроса:

1. если вложенный запрос возвращает пустой столбец результатов, то проверка ALL имеет значение TRUE. Считается, что условие сравнения выполняется, даже если результаты вложенного запроса отсутствуют.
2. если операция сравнения дает результат TRUE для каждого значения в столбце, то проверка ALL возвращает значение TRUE. Условие сравнения выполняется для каждого значения, полученного вложенным запросом.
3. если операция сравнения дает результат FALSE для какого-нибудь значения в столбце, то проверка ALL возвращает значение FALSE. В этом, случае можно утверждать, что условие поиска выполняется не для каждого значения, полученного вложенным запросом.
4. если операция сравнения не дает результат FALSE ни для одного значения в столбце, но для одного или нескольких значений дает результат NULL, то проверка ALL возвращает значение NULL. В этой ситуации нельзя с определенностью утверждать, для всех ли значений, полученных вложенным запросом, справедливо условие сравнения; может быть, для всех, а может и нет – все зависит от "настоящих" значений неизвестных данных.

8.3.3 Вложенные запросы и объединения

При чтении данной главы вы, возможно, заметили, что многие запросы, записанные с применением вложенных запросов, можно также записать в виде многотабличных запросов. Такое случается довольно часто, и SQL позволяет записать запрос любым способом.

8.3.4 Уровни вложенности запросов

Все рассмотренные до сих пор запросы были "двухуровневыми" и состояли из главного и вложенного запросов. Точно так же, как внутри главной запроса может находится вложенный запрос, внутри вложенного запроси может находиться еще один вложенный запрос.

8.3.5 Вложенные запросы в предложении HAVING

Хотя вложенные запросы чаще всего применяются в предложении WHERE их можно использовать и в предложении HAVING главного запроса. Когда вложенный запрос содержится в предложении HAVING, он участвует в отборе группы строк.